不打开前端但打开测试环境：make localtestnv

Data.go, query

# 启动docker

## 命令行式启动

* 进入docker目录, 即运行:   
  cd go/src/bitbucket.org/applysquare/django/docker/development
* 在bash中运行:   
  docker-machine start default # 按默认设置启动docker  
  docker-machine env # (Optional)重设IP之类  
  docker-compose up -d # 启动docker中的container (稍等会再运行该命令)
* 如果不能运行, 则运行:  
  make reboot
* 如果还不能运行, 在根目录上运行如下命令:  
  cd ~ # 可以不切换目录, 当前development目录下即可  
  eval $(docker-machine env default) # 配置某些环境变量 (可在 vi .bash\_profile 里作全局设置)
* 打开网页: 192.168.99.100:5984/\_utils, 如果正常(会有一些延迟), 说明docker正常启动
* 可以用以下命令, 查看docker运行的状态:  
  docker-compose ps

## 命令行式退出

* 在命令行中运行:  
  docker-machine stop default

## 普通方式退出

* 在docker中: ctrl+d (临时退出)

# 在docker中测试数据

* 进入以下目录:  
  cd go/src/bitbucket.org/applysquare/applysquare-go/pkg/site/hwxr
* 在bash中运行下列命令:  
  go test -v -args --logtostderr --data=realdata( or testdata, 用的哪个数据, pgweb就会导入哪个)
* 如果报错 panic: Set A2\_TESTING\_ADDR env to your docker IP, 则运行下列命令即可:  
  export A2\_TESTING\_ADDR=192.168.99.100
* 如果报错超时: \*\*\* Test killed with quit: ran too long (10m0s). 可以在go test时添加timeout的参数, 100m代表100分钟, 0代表无限制 (0可能要加单位):  
  go test -v -timeout=100m -args --data=realdata –logtostderr  
  -args前面是go的默认flag, 后面是自定义的flag

# 数据上传

上传之前, 确保数据都测试通过

* 切换到目录:  
  cd go/src/bitbucket.org/applysquare/applysquare-go
* 在bash中运行:  
  make generate # 生成相关的配置文件
* 没有什么错误的话, 再运行:  
  git status # 查看文件状态(通常queryspec.go, .DS\_Store会有修改)  
  git add pkg/site/hwxr/dataset/queryspec.go # 将特定文件加载到staged区  
  git commit -m ‘modify query for school GPS’ # 提交修改  
  git push # 将修改上传到服务器
* 几分钟后, 到网站后台查看提交记录; 如果OK则可以提一个MR

# Pgweb启动

## 启动

* 先关闭docker
* 点击桌面的pgweb\_darwin\_386 (或64位的)
* 会自动跳转, 打开网址:   
  localhost:8081
* 启动docker (详见section 1)
* 输入服务器地址, connect即可:   
  postgres://postgres:postgres@192.168.99.100:5432/a2\_go\_test?sslmode=disable

# Git文件

## clone整个project

* 切换至目录:  
  cd go/src/bitbucket.org/applysquare
* 运行 (从<https://ma.applysquare.net/eng/go下>, 选择SSH下对应的地址):  
  git clone [git@ma.applysquare.net:eng/go.git](mailto:git@ma.applysquare.net:eng/go.git)

## 获取某一个branch

* 直接新建一个branch:  
  git branch ccsstest
* 再将远程数据pull下来即可:  
  git pull # 若失败, 按GIT提示处理即可

## 上传本地分支

* 如果是首次git push, 可能会说no upstream branch, 按提示运行下列命令即可:  
  git push –set-upstream origin atatest

## 获取远程分支

方法一:

* 直接git checkout -b <branch>, 再git pull, 然后按提示运行下列命令即可:  
  git branch –set-upstream-to=origin/<remote\_branch> <local\_branch>

方法二:

* git clone -b <remote branch> <git url>

# Graph*i*QL本地测试

## 导入数据

GraphiSQL使用的是本地的真实数据, 因此:

* 切换目录, 并pull最新数据:  
  cd go/src/bitbucket.org/applysquare/applysquare-go/realdata  
  git pull
* 如果没有对应项目 (如CCSS) 的数据目录, 则:  
  git clone [git@ma.applysquare.net:data/ccss.git](mailto:git@ma.applysquare.net:data/ccss.git)
* 如果数据结构有变化, 如增减表之类的, 要先:  
  cd go/src/bitbucket.org/applysquare/applysquare-go  
  make generate
* 切换目录, 并make index:  
  cd go/src/bitbucket.org/applysquare/django/docker/development  
  make index\_ccss
* 如果报错\*\*\* No rule to make target ‘index\_ata’, 则运行git pull确保代码最新

## 测试

打开浏览器, 输入: 192.168.99.100:8000/devproxy/graphiql

## 注意

Make index是导入的当时的数据及queryspec.go, 若是数据变动, 则需要再次make index

# Windows中访问github

## 安装git

下载windows安装包, 按步骤安装即可. 注意: 最后的时候, 要将git路径加至全局变量中.

## 生成ssh

* 在Git路径, 找到git-bash.exe, 打开后运行以下命令:  
  ssh-keygen -t rsa -b 4096 -C "feifei\_pc" # 将feifei\_pc换成对应自己的名字  
  会生成一个ssh码之类的文件, 中间的确认一直按回车即可
* 通常, 默认情况下, 该文件在以下目录:  
  c:\Users\Administrator\.ssh\id\_rsa.pub

## 添加ssh key

* 将文件中的值, 拷内至 <https://ma.applysquare.net/profile/keys> 即可. (该例中, Title它会自动识别为feifei\_pc), 添加即可.

# Mac中git pull失败

在网络及无其他问题的情况下, git pull报如下错误:

* Permission denied (publickey)  
  fatal: Could not read from remote repository
* 或者出现弹窗 “Enter your password for the SSH key “id\_rsa”

说明ssh出现问题, 重新生成并添加在github账户里即可. 步骤如下:

1. 生成Key (后面的**-b 4096**不要忘记):  
   ssh-keygen -t rsa -C "your.email@example.com" -b 4096
2. 将id\_rsa.pub里的内容添加到github 🡪 Setting 🡪 SSH Keys里即可, title可以写为一个可标识的名字, 详见Section 7.

更多详见<https://ma.applysquare.net/help/ssh/README>.

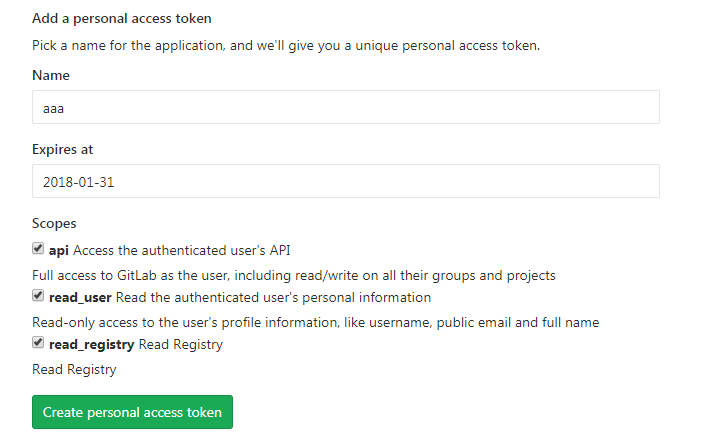
# 版本升级

## Go升级

可直接下载mac的安装包, 双击安装即可, 默认即会覆盖旧版本. 完成后在命令行运行go version来检查当前版本号

## Docker升级

升级步骤如下 (见 <https://ma.applysquare.net/eng/docker/container_registry>):

* 切换到development目录:  
  cd go/src/bitbucket.org/applysquare/django/docker/development
* 若已开通2步验证, 先到Settings -> Access Tokens 生成一个新的token, 该token作为密码登录:  
    
  docker login ma.applysquare.net:4567

登录名

* 一般几秒钟即可OK, 时间太长通常是超时timeout, 重启docker:  
  docker-machine restart  
  docker-compose up -d  
  docker-compose ps
* 登录成功后, pull代码即可:  
  docker-compose pull

# 部署本地CCSS环境

如果本地可用 (不要make fuse), 就不要升级/更新代码, 直接启动即可.

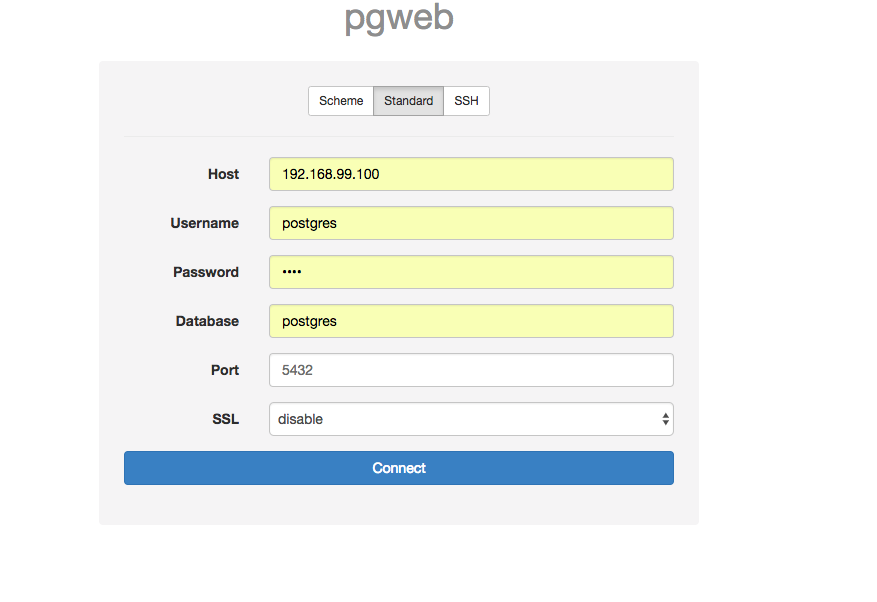
步骤较复杂, 需要特别的权限 (先确保docker是打开的):

* 切换目录:  
  cd go/src/bitbucket.org/applysquare/applysquare-go
* 现在有v1, v2两个版本, 切换到v1:  
  cd js/a2-main/v1
* 用以下命令安装可能要的package:  
  yarn # 没有新包就不用该命令  
  如果没有权限, 向技术部负责人申请即可
* 如果报错node版本太低, 先check当前可用版本 (具体版本咨询技术):  
  nvm list  
  nvm install 8.1.3 # 如果没有相应版本, 安装特定版本  
  nvm use 8 # 如果有, 该命令可以临时切换版本
* 切换到v2, 用样yarn, nvm相关操作
* 切换回到applysquare-go目录, 如果前端有修改, 则:  
  git checkout master  
  git pull  
  git checkout <current\_branch>  
  git merge --no-ff master # 合并master的相关代码  
  make fuse

下面是导入数据的部分:

* 如果skima有变动, 则执行下面的代码:  
  make generate
* 最后, 导入数据即可:  
  cd go/src/bitbucket.org/applysquare/django/docker/development  
  ~~make index\_ccss # 导入的是realdata的数据~~   
  ./run\_indexer .sh runonce ccss # 导入的是realdata的数据

[小飞机需要先关掉], 打开 ccss.local.applysquare.net:8000/analyze/colleges 即可 (可能有几分钟的延迟), 可能需要用Ctrl + Shift + R强制刷新 (清除缓存).

本地部署的对应数据库可以访问:  


如果失败:

* 检查docker状态 (development下):  
  docker-compose ps
* 如果docker可用, 仍不行, 重启试一下:  
  make reboot
* 再检查服务器状态, 看是否有错误信息:  
  docker-compose logs -f goserver
* 如果有, 看看是否需要给go升级  
  brew list # 检查go是否通过brew安装, 如果不是, 直接下载安装包升级即

# 备注

## CCSS

Pkg/site/ccss/dataset/data.go中的enumGradeMap在后台management/manager\_survey.go中引用, 而django中引用外部变量必须大写.

# Apache Drill错误记录

* Error: SYSTEM ERROR: NumberFormatException  
  如果报错的数据类型为int, 可能是drill将空值转换为int时报错, 可以将option中的drill.exec.functions.cast\_empty\_string\_to\_null 改为true; 如果报错的数据类型本身为string, 查看原始数据, 是否有空文件